package com.twitter.cr\_mixer.featureswitch

import com.twitter.abdecider.LoggingABDecider

import com.twitter.abdecider.UserRecipient

import com.twitter.cr\_mixer.{thriftscala => t}

import com.twitter.core\_workflows.user\_model.thriftscala.UserState

import com.twitter.discovery.common.configapi.FeatureContextBuilder

import com.twitter.featureswitches.FSRecipient

import com.twitter.featureswitches.UserAgent

import com.twitter.featureswitches.{Recipient => FeatureSwitchRecipient}

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.product\_mixer.core.thriftscala.ClientContext

import com.twitter.timelines.configapi.Config

import com.twitter.timelines.configapi.FeatureValue

import com.twitter.timelines.configapi.ForcedFeatureContext

import com.twitter.timelines.configapi.OrElseFeatureContext

import com.twitter.timelines.configapi.Params

import com.twitter.timelines.configapi.RequestContext

import com.twitter.timelines.configapi.abdecider.LoggingABDeciderExperimentContext

import javax.inject.Inject

import javax.inject.Singleton

/\*\* Singleton object for building [[Params]] to override \*/

@Singleton

class ParamsBuilder @Inject() (

globalStats: StatsReceiver,

abDecider: LoggingABDecider,

featureContextBuilder: FeatureContextBuilder,

config: Config) {

private val stats = globalStats.scope("params")

def buildFromClientContext(

clientContext: ClientContext,

product: t.Product,

userState: UserState,

userRoleOverride: Option[Set[String]] = None,

featureOverrides: Map[String, FeatureValue] = Map.empty,

): Params = {

clientContext.userId match {

case Some(userId) =>

val userRecipient = buildFeatureSwitchRecipient(

userId,

userRoleOverride,

clientContext,

product,

userState

)

val featureContext = OrElseFeatureContext(

ForcedFeatureContext(featureOverrides),

featureContextBuilder(

Some(userId),

Some(userRecipient)

))

config(

requestContext = RequestContext(

userId = Some(userId),

experimentContext = LoggingABDeciderExperimentContext(

abDecider,

Some(UserRecipient(userId, Some(userId)))),

featureContext = featureContext

),

stats

)

case None =>

val guestRecipient =

buildFeatureSwitchRecipientWithGuestId(clientContext: ClientContext, product, userState)

val featureContext = OrElseFeatureContext(

ForcedFeatureContext(featureOverrides),

featureContextBuilder(

clientContext.userId,

Some(guestRecipient)

)

) //ExperimentContext with GuestRecipient is not supported as there is no active use-cases yet in CrMixer

config(

requestContext = RequestContext(

userId = clientContext.userId,

featureContext = featureContext

),

stats

)

}

}

private def buildFeatureSwitchRecipientWithGuestId(

clientContext: ClientContext,

product: t.Product,

userState: UserState

): FeatureSwitchRecipient = {

val recipient = FSRecipient(

userId = None,

userRoles = None,

deviceId = clientContext.deviceId,

guestId = clientContext.guestId,

languageCode = clientContext.languageCode,

countryCode = clientContext.countryCode,

userAgent = clientContext.userAgent.flatMap(UserAgent(\_)),

isVerified = None,

isTwoffice = None,

tooClient = None,

highWaterMark = None

)

recipient.withCustomFields(

(ParamsBuilder.ProductCustomField, product.toString),

(ParamsBuilder.UserStateCustomField, userState.toString)

)

}

private def buildFeatureSwitchRecipient(

userId: Long,

userRolesOverride: Option[Set[String]],

clientContext: ClientContext,

product: t.Product,

userState: UserState

): FeatureSwitchRecipient = {

val userRoles = userRolesOverride match {

case Some(overrides) => Some(overrides)

case \_ => clientContext.userRoles.map(\_.toSet)

}

val recipient = FSRecipient(

userId = Some(userId),

userRoles = userRoles,

deviceId = clientContext.deviceId,

guestId = clientContext.guestId,

languageCode = clientContext.languageCode,

countryCode = clientContext.countryCode,

userAgent = clientContext.userAgent.flatMap(UserAgent(\_)),

isVerified = None,

isTwoffice = None,

tooClient = None,

highWaterMark = None

)

recipient.withCustomFields(

(ParamsBuilder.ProductCustomField, product.toString),

(ParamsBuilder.UserStateCustomField, userState.toString)

)

}

}

object ParamsBuilder {

private val ProductCustomField = "product\_id"

private val UserStateCustomField = "user\_state"

}